VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

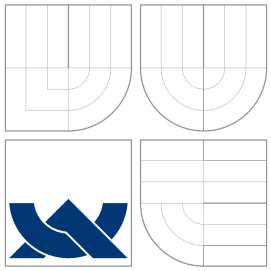
NÁSTROJ PRO SPRÁVU ZOBRAZOVACÍCH ZAŘÍZENÍ V SYSTÉMU GNU/LINUX

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

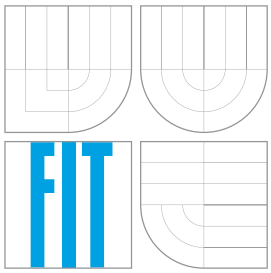
AUTOR PRÁCE
AUTHOR

PETR KLANG

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

NÁSTROJ PRO SPRÁVU ZOBRAZOVACÍCH ZAŘÍZENÍ V SYSTÉMU GNU/LINUX

GNU/LINUX ADMINISTRATION TOOL FOR ATTACHED DISPLAY DEVICES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — PETR KLANG
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN KNĚŽÍK
SUPERVISOR

BRNO 2012

Abstrakt

Cílem této práce je vytvoření aplikace pro správu monitorů v systému GNU/Linux, která by dokázala uložit nastavenou konfiguraci a později toto nastavení vyvolat. Při připojení nového monitoru vyhodnotí všechny konfigurace a vybere tu, která nejlépe vyhovuje. Aplikace se spouští po startu manažera obrazovky. Práce se také zabývá výběrem vhodného prostředí, popisem implementace a testováním.

Abstract

The aim of this thesis is to create an application for monitor administration in GNU/Linux system which would be able to save the set configuration and recall that setting later. When a new monitor is connected, the application evaluates all configurations and chooses the most convenient one. The application is run after the screen manager has been started. The thesis is also concerned with the selection of suitable tools, description of implementation, and testing.

Klíčová slova

X11, X server, XRandr, monitor, linux, Ubuntu

Keywords

X11, X server, XRandr, monitor, linux, Ubuntu

Citace

Petr Klang: Nástroj pro správu zobrazovacích zařízení v systému GNU/Linux, bakalářská práce, Brno, FIT VUT v Brně, 2012

Nástroj pro správu zobrazovacích zařízení v systému GNU/Linux

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jana Kněžíka. Dále prohlašuji, že jsem použil jen uvedených pramenů a literatury a neporušil autorská práva.

.....
Petr Klang
15. května 2012

Poděkování

Děkuji panu Ing. Janu Kněžíkovi za odbornou pomoc, pečlivé vedení a poskytování rad a materiálů.

Obsah

Kapitola 1

Úvod

Doba, kdy jsme si vystačili pouze s jedním monitorem a rozlišením 800×600 je již dávno pryč, šla dokonce ještě dál a spousta lidí si nedokáže představit, že by měli pouze jeden monitor. Notebook, jakožto mobilní počítač, se stává žádanější, než klasický počítač. Prodávají se ještě menší zařízení jako například netbook. Při práci na počítači potřebujeme vidět více ploch zároveň. *Externí monitor* nám umožňuje více místa pro jednotlivé programy.

Ergonomie notebooku není stavěná pro práci každý den a ač se přenosné počítače kupují místo pracovních stanic, není práce na nich ideální. Notebook je stavěn s ohledem na mobilitu, takže pouhé postavení na pracovní stůl z něj nedělá dobrý pracovní nástroj. Mnoho pracovníků toto řeší připojením další pracovní plochy, nahrádního monitoru za vestavěný v notebooku. Problém ale nastává při přenášení tohoto počítače. Nastavení, které je na jednom pracovním stole nemusí být uplatnitelné u jiného. Dalším, mnohem závažnějším důvodem je návrat pracovníka ke stejnému stolu. Je pravděpodobné, že bude chtít mít monitory nastavené stejně, aniž by vše musel pokaždé složitě nastavovat.

Při připojení dataprojektoru také vyžadujeme, aby jeho nastavení bylo co nejrychlejší a nejpohodlnější. Prezentování na dataprojektoru je velice běžná praxe, jak v rámci vysokých škol tak firem, ale správné nastavení často může trvat déle než samotná prezentace.

Hardwarové prostředky jsou stále dostupnější, součástky se zlevňují a my si můžeme dovolit mít více monitorů. Kde ale často narážíme jsou prostředky softwarové. Abychom nemuseli trávit dlouhý čas nastavováním externích monitorů, jsou nám k dispozici různé grafické programy. Ne všechny programy přinášejí možnosti, které bychom potřebovali. Zvlaště pak jakési uchovávání jednou nastavených hodnot.

V komerční sféře jsou tyto programy často integrovány přímo v systému. Například Windows 7 podporuje rychle přepínání připojených monitorů za pomoci klávesové zkratky. V sféře svobodných operačních systémů je situace poněkud horší, programy jsou různorodé a ne vždy plní takovou funkci, jakou bychom si představovali. Rozhodl jsem se tedy zaměřit se právě na tuto sféru a zjistit jaké jsou zde možnosti.

Mojí motivací bylo prostudovat možné prostředky pro implementaci takové aplikace, která zajistí v GNU/Linux vytváření konfiguračních profilů s různým nastavením připojených monitorů. Program bude detekovat připojené monitory při spuštění *display manageru* a bude umožňovat široké množství nastavení monitorů. Možná se ptáte proč? Linux má mnoho programů pro nastavování monitorů, ale co postrádá je ukládání konfigurace. Proto bude napsaná aplikace umožňovat ukládání aktuálního nastavení a její zpětné vyvolání. Zároveň bude aplikace tyto konfigurace spravovat, popřípadě se rozhodovat jaká uložená konfigurace bude použita.

V následujících kapitolách Vám vysvětlím, jaké prostředky jsem použil pro vývoj aplikace. Na dalších stránkách rozebírám v jakém prostředí byla aplikace napsána. Další kapitolu jsem věnoval X serveru a základním informacím. Také je zde vysvětlena některá terminologie, která je s X serverm spjata. Nadále se věnuji konceptu programu, jak vypadá jeho návrh a které komponenty obsahuje. Poté popisuji, jak se konfigurace ukládají a jaký mají tvar. Dle mého je důležité informovat, jak spolu jednotlivé aplikace komunikují a jaké mají definové rozhraní. Následně se zmiňuji o automatickém spouštění aplikací a napsání vhodného skriptu. K závěru uvádím, jak vypadá klientská část programu, jakou obsahuje funkcionalitu a jak je možné aplikaci nainstalovat. Na úplný závěr se věnuji testování aplikace a její funkcionalitě na rozdílných strojích. Poslední kapitola je pak shrnutí všeho, čeho jsem se naučil, jestli se povedlo splnit zadání a jaké jsou možnosti pokračování v tomto projektu.

Kapitola 2

Použitá výchozí technologie

Na záčátku jsem se musel rozhodnout v čem bude výsledný program napsaný. Je jasné, že musí být psaný pod GNU/Linux, ale nabízí se dvě varianty, jak k monitorům přistupovat. Volil jsem mezi prostředky, které jsou běžně dostupné a rozšížené.

Jako první v úvahu připadlo *VESA Bios Extended*. Většina grafických karet má v sobě BIOS [1], toto rozšíření od VESA[11] umožňuje nastavit rozlišení u grafické karty, aniž bychom potřebovali ovladače. Problém ovšem je, že ne každá grafická karta má toto rozšíření. Také ne vždy grafická karta umožňuje takto nastavit všechny monitory, ale třeba pouze jeden, jako „záchranný mód“. Přínosem by mohlo být, že není potřeba žádné grafické nástavby. Bohužel funguje pouze s právy root a je nutné použít systém x86 nebo jeho emulaci. Poslední verze 3.0 vyšla v roce 1998.

VESA Bios Extended

Nevýhody

- maximální rozlišení definované standardem 1600×1200
- funguje pouze na systémech x86
- ke správnému fungování jsou nutné privilegia root
- není implementováno ve všech grafických kartách

Výhody

- není potřeba mít naintalovaný žádný ovladač
- není nutno mít nainstalovaný X Window systém

Další možností, která se nabízela je využít, to co nabízí X server, který je téměř v každé desktopové distribuci linuxu. Jedno z jeho rozšíření je RandR[6], což znamená změnu velikosti, otočení a zrcadlení[2], bylo přidáno v roce 2001. Od té doby přešlo do verze 1.31. Tento doplněk ulehčuje práci s monitory. Předtímto zlepšením, bylo vždy, když jste potřebovali změnu rozšíření nebo otočení monitoru restartovat celý systém X. Díky RandR je možné tyto změny řešit bez restartování, dynamicky, za běhu serveru.

[1] Basic input output system
[2] Resize, Rotate and Reflect

RandR

Nevýhody

- X11 mívají problémy s ovladači

Výhody

- X window systém je ve většině linuxových distribucí
- umožňuje různé varianty nastavení

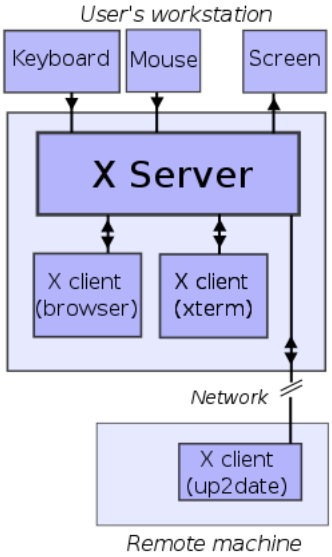
VESA nabízí přístup na nižší úrovni a není za potřeba jakýchkoliv ovladačů, či přídavného software. Problém nastává, že je tato možnost poměrně zastaralá a vývoj v oblasti monitorů od roku 1998 pokročil. Také při zkoušení této varianty se mi nepovedlo zprovoznit nastavení více než jednoho monitoru. Druhá možnost obsahuje vyšší míru abstrakce, na této vrstvě nás nezajímá od jakého výrobce je grafická karta, protože mezi XRandrem a hardwarem stojí X server, který tyto ovladače zajišťuje. Pro náš případ jsem zvolil programování s X serverem a zmíněným rozšířením.

Jakožto operační systém jsem zvolil Ubuntu, které má nakročeno k širokému spektru uživatelů. Samozřejmě, že systém není nejdůležitější volba, protože se jedná o svobodný software a použité technologie jsou doinstalovatelné i do jiných distribucí. Jde spíše o to, jaké prostředky se využijí s přihlídnutím na výchozí možnosti, které Ubuntu nabízí.

Kapitola 3

X window system

Okenní systém X, nebo-li X a X11, je základním grafickým prostředím snad každé linuxové distrubuce. X systém funguje obdobně jako síťové aplikace, definuje nám, jak vypadá server a jakým protokolem komunikuje s klientem. Server zajišťuje nejnižší vrstvu, pomocí ovladačů pracuje s vstupními a výstupními zařízeními. O každé změně, která proběhne u těchto zařízení, informuje klienty. Pokud se tedy pohne myší, dá o tom server vědět klientům, stejně tak stisku klávesy apod. Klient se musí rozhodnout, jestli daná informace je pro něj podstatná a zachovat se podle toho. Klient je v tomto případě okno, neboli grafická aplikace. Komunikace probíhá přes pakety, které se snadno dají využít například při přenosu po síti pomocí TCP/IP.

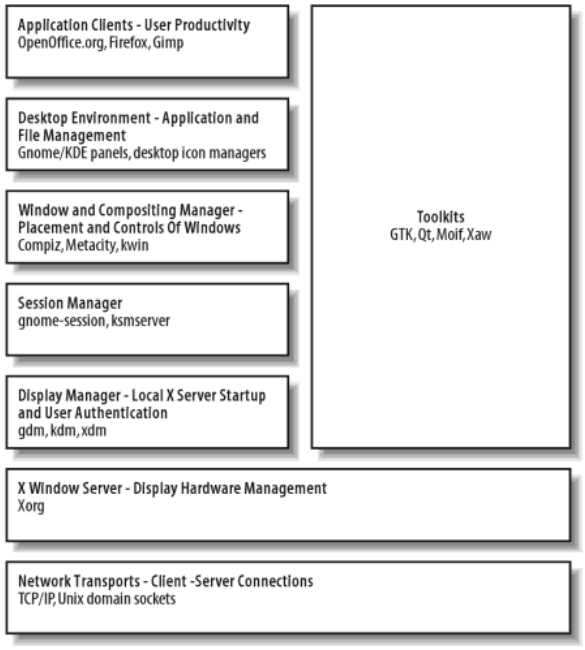


Obrázek 3.1: Architektura X serveru

X protokol je to nejpodstatnější na systému X. Postupně se ale na tento protokol navázalo mnoho aplikací, implementací serverů, klientů a knihoven. Důležitou knihovnou je Xlib, ta pomáhá odstínit X protokol tak, aby programátor mohl srozumitelněji psát grafické aplikace. Je psaná v jazyce C a je pouze základním kamenem, který umožnil další nádstavby a knihovny, protože programátor musí stále řešit spoustu technickým detailů. Proto vznikly knihovny nebo takzvané toolkity, které grafickou práci zjednodušují a zpříjemňují. Jako příklad bych uvedl Qt nebo GTK, díky nimž jsou napsané desktopy KDE a GNOME.

3.1 Historie

Systém X vznikl již v roce 1984 v Massachusettském technologickém institutu, kladl za cíl sjednocení grafického prostředí, v té době velice roztříštěného. Od toho roku do roku 1987 vzniklo jedenáct verzí systému. Systém X11 tedy se svojí verzí vydržel až do současnosti. Neznamená to však, že by byl systém neměnný, systém se rozšiřuje o různá rozšíření a mění se číslo verze, princip fungování, ale zůstává stejný. Zakladatelé si stanovili několik zásad, kterých se drželi a díky nimž se systém spíše rozšiřoval.



Obrázek 3.2: Jednotlivé vrsty používané v X Window Server[9]

3.2 Zásady návrhu

Bob Scheifler a Jim Gettys stanovili principy X takto (uvedeno v Scheifler/Gettys 1996)[8]

- Nové funkce přidávat pouze v případě, kdy vykonavatel není schopen dokončit aplikace bez nich.
- Je stejně důležité rozhodnout, na co se systém bude používat, jako na co se používat nebude. Raději, než zcela univerzálně použitelný systém, vytvořit systém rozšiřitelný, aby další potřeby mohli být plněny vzestupně kompatibilním způsobem.
- Jediná věc, horší než zobecňovat z jednoho příkladu, je nepoužití příkladu vůbec.
- Pokud problému není zcela porozuměno, je pravděpodobně nejlepší neposkytovat žádné řešení.
- Můžete-li získat 90 % požadovaného účinku za 10 % práce, použijte jednodušší řešení.
- Omezit složitost na minimum

První zásada byla upravena během návrhu X11 na znění: Nové funkce přidávat pouze v případě, pokud víte o nějaké aplikaci, která je bude vyžadovat.

I přestože jednou ze zásad bylo omezení složitosti na minimum, stal se X window systém „molochem“, který nabízí spoustu možností, ale jeho architektura je zastaralá a 30 let neměnná. Rozšiření, které nad systémem vzniklaly umožňovaly, abychom ho mohli používat až do současnosti, ale jádro je opravdu staré, komunikace server-klient byla stavěná pro možnost síťové komunikace, práce s fonty apod. To vše ve stylu 80. let, což je již téměř 30 let. Také se trochu vymyká filozofii linuxových programů – Je lepší stvořit malý program, který bude dobře fungovat a bude snáze opravitelný, než velký program, kterému nikdo neporozumí.

Do Linuxu se implementace X serveru dostala především za pomoci *XFree86*, projekt, který začal v roce 1992[16], avšak kvůli změně licence na nesvobodnou došlo k rozdvojení a v roce 2004 přešly linuxové distribuce na *X.org*[17].

„Xorg je veřejná open-source implementace systému X window verze 11. Jelikož je Xorg nejpopulárnější volbou mezi uživateli Linuxu, jeho všudypřítomnost vedla k tomu, že se stal základním předpokladem pro provozování grafických aplikací a byl tak masově adoptován většinou distribucí.“[5]

3.3 X terminologie

Pro pochopení některých principů je důležité si úvest, co znamenají některá anglická slova z pohledu X11. Není tím myšleno, že by český jazyk neměl stejný ekvivalent, ale překladem bychom mohli částečně ztratit význam.

Display Tento pojem se používá pro kolekci více monitorů mající jednu klávesnici a jednu myš. Dalo by se říci, že pokud počítač ovládá jeden člověk, má k dispozici jeden *display* a je jedno kolik monitorů, myší a klávesnic vlastní. Vše ovládá pomocí tohoto jednoho sezení, jednoho `display`.

Hostname Název počítače, pokud se jedná o lokální počítač, tak název není nutný.

Screen Každý monitor, který má vlastní soubor oken je označen jako screen a má vlastní číslo *screennumber*. Povětšinou platí, že jedna grafická karta má jeden *screen*.

Output Česky řečeno výstup. Jedná se o přípojku ke které monitor můžeme připojit, na ten však nemusí být připojený žádný monitor.

Root Window Hlavní, kořenové okno, okna se v X tvoří jako podokno nadřazeného okna, podobně jako procesy v linuxu. Toto okno zároveň slouží jako pozadí oken a určuje tak maximální velikost.

Display manager Program, který se stará o přihlášení k *display*, autorizaci a často jde o první spouštěný program v X11.

Framebuffer Výstupní zobrazovací zařízení grafických karet, podle barevné hloubky je generován patřičný počet bitů na jeden pixel.[14]

Mode Skupina informací o rozlišení a frekvenci, *screnn* má skupinu těchto módů. Monitor má také skupinu těchto módů, které se pro něj mohou používat.

Crtc Kontroler, který se stará o monitor, posílá mu obraz z framebufferu grafické karty. Každý monitor by měl mít přiřazen jeden, pokud se nejedná o klonování, kdy při zachovaní stejného módu může být použit jeden kontroler na dva a více monitorů.

Kapitola 4

Použití knihovny XRandR

Jako vhodnější variantu jsem zvolil knihovnu a rozšíření X Serveru RandR. Knihovna nabízí několik funkcí pro práci s monitory a jejich kontrolery (*crtc*), umožňuje okamžité nastavování několika vlastností monitoru bez znovu spouštění X serveru. Také nabízí nastavení *screen*. Zde jsou bohužel možnosti knihovny omezené. Pokud například máme více grafických karet, mužeme se setkat s tím, že budeme chtít nastavit rozložení těchto grafických karet, dle rozložení na stole. To bohužel knihovna Xrandr nenabízí. Tyto konfigurace jsou nutné nastavit v konfiguračním souboru Xorg.conf, kde mužeme nastavit maximální velikost, ale také pozici všech připojených monitoru jako celku. Mužeme tedy říct, jestli monitory grafické karty jsou vlevo, resp. vpravo, nahore či dole, od druhé grafické karty.

V novejších verzích X serveru již konfigurační soubor `Xorg.conf`[3] nenaleznete v `/etc/X11/Xorg.conf`, protože tyto konfigurace jsou automaticky generované a soubory, které jsou potřeba explicitně specifikovat se nacházejí v adresáři `/usr/share/X11/Xorg.conf.d/`[1]. Není ale problém tento soubor vytvořit.

```
xklang00@merlin: ~$ X -configure
```

```
xklang00@merlin: ~$ cp Xorg.conf /etc/X11/Xorg.conf
```[2]

A následně dopsat do něj pouze potřebné parametry. Soubor `/etc/X11/Xorg.conf` má vyšší prioritu a použije se konfigurace z tohoto souboru. [3]

4.1 Duležité funkce

Xrandr umožnuje nastavovat *screen*, jak již bylo řečeno, nastavuje šířku a výšku v pixelech i v milimetrech. Dále umožnuje nastavovat monitory a to známými způsoby – měnit rozlišení a frekvenci dle nadefinovaných módu viz výše 3.3 a otočení. Také tyto módy přidávat a odebírat je, ale ne všechny takto přidané módy musí monitor podporovat. S otočením se dá konfigurovat také zrcadlení podle osy x i y.

V našem případe nejduležitejší vlastností je možnost monitory klonovat, nebo je zobrazovat vedle sebe. To probíhá za pomoci pozice kontroleru monitoru a framebufferu, framebuffer vygeneruje podle rozlišení celého *screen* patřičné bity a předá je kontrolerům podle pozice,

[1] Pro distribuci Ubuntu

[2] Je však nutné mít spuštěný X server.

[3] Na testovacím stroji, který mi byl poskytnut v laboratori S214, bylo například nutné překonfigurovat používané ovladače na grafickou kartu od ATI a nastavit pozici *screen*.

odkud se mají vykreslovat a rozlišení monitoru. Pokud tedy jeden kontroler má pozici `[0,0]` a rozlišení `1366×768` a druhý pozici `[1366,0]` a rozlišení `1024×768`, bude se plocha druhého monitoru zobrazovat vpravo od prvního. Pokud tedy monitory mají stejnou pozici, bude se vykreslovat společná část. Pokud mají zároveň i stejné rozlišení můžeme mluvit o klonování. Pokud je u klonování stejný mód monitrů, může být použit jeden *crtc* pro oba monitory.

Ve verzi 1.3 dostává knihovna podporu také méně známého panningu a scalingu. Panning, což bychom mohli do češtiny přeložit jako posouvání, umožnuje zobrazit na danné rozlišení rozlišení vetší a to za pomoci posouvání obrazu. Pokud se myší posuneme na konec obrazovky, obrazovka se podle zvolené velikosti posune. Mužeme tak poměrně elegantně zobrazit na monitoru s malým rozlišením monitor s rozlišením větším. Scaling, který bychom mohli česky vyjádřit jako „měnění měřítka“, nebo lépe škálování. Umožňuje zvětšovat transformaci zobrazení pomocí transformační matice. Ta by měla umožnovat nejen zvětšení a zmenšení obrazu, ale také například rotaci.[4]

Xrandr také rozšiřují události, které vyvolává X server. Nově zprostředkovává posílání a přijímání signálů, když se změní *screen*, *output* i *crtc*. Také nabízí zjištění vlastností monitoru, kterými se monitor identifikuje a kterými dává o sobě znát. Toho jsem využil pro zjištení EDID. EDID je unikátní identifikátor monitoru[13], který obsahuje informace o sériovém čísle, výrobci, ale také o možných rozlišeních, respektivě módech.

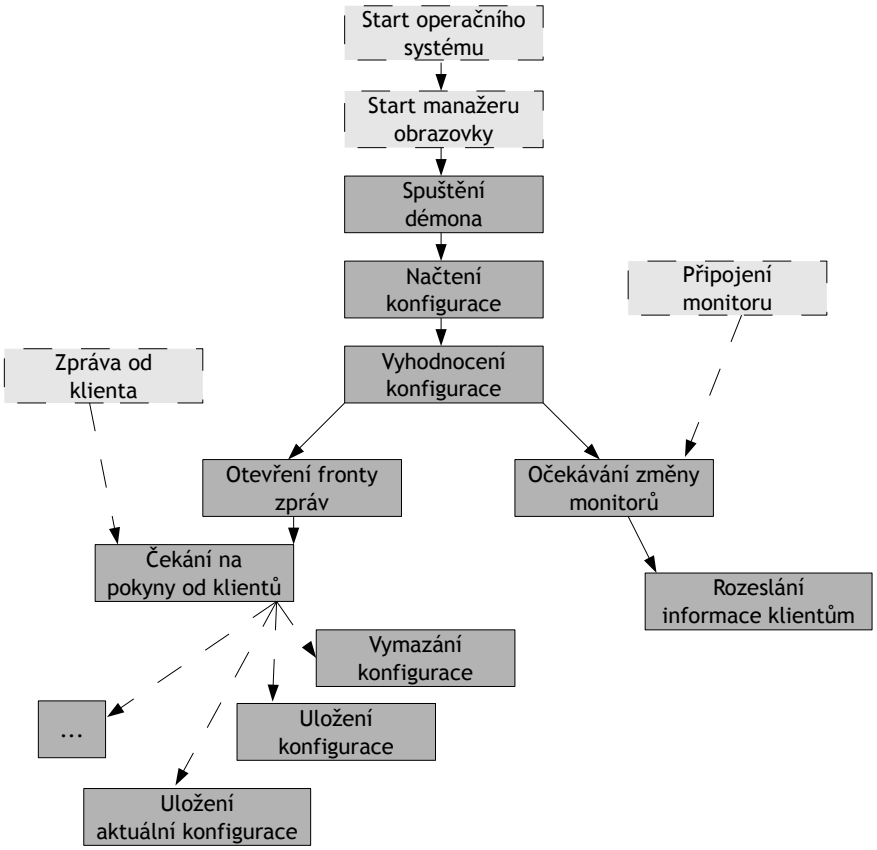
Pro programování s knihovnou Xrandr je nutné nainstalovat knihovnu `libxrandr-dev` a překládat s parametrem `-lrandr`.

[4] V praxi na testovacím počítači se otočení nepodařilo zprovoznit, ale naštěstí lze rotaci provádet i jinak, viz rotace.

Kapitola 5

Koncept projektu

Program byl navržen tak, aby z velké části fungoval automaticky a nepotřeboval zásah uživatele. Jádrem programu je deamon, který beží na pozadí systému. S tímto deamonem mohou komunikovat jednotliví klienti a to na základě předdefinovaného protokolu. Od uživatele se vyžaduje, aby si podle svých preferencí nadefinoval pro jednotlivé unikátní připojené monitory konfiguraci, která mu nejvíce vyhovuje. Program tuto konfiguraci načte a při opětovném připojení stejné kombinace monitorů bude použit první vyhovující. Uživatel tedy jednou monitory nastaví a pak se již nemusí starat o to, aby se monitory připojily stejně.



Obrázek 5.1: Návrh programu

5.1 Deamon

Pro pořádek bych rád vysvětlil, co je to deamon a co by měl program s takovou charakteristikou splňovat. Česky bychom mohli říci démon. Je to program který běží na pozadí linuxového operačního systému a zajišťuje často danou konkrétní službu. Například `nfsd` starající se o fungování Network File System. Písmenko `d` na konci značí, že se jedná o démona. Takovýto systémový proces musí být nezávislý od ostatních pocesů běžících v systému. Je tedy sepsáno několik pravidel, podle kterých se postupuje při vytváření démona:[12]

- Odvázání se od rodičovského procesu, démon je převzat procesem `init`[1]
- Změnění práv procesu – `umask`[2]
- Vytvoření Session ID
- Stání se vedoucím skupiny procesů[3]
- Změnění pracovního adresáře na kořenový adresář s `/`[4]
- Uzavření všech otevřených souborů, včetně standardního vstupu a výstupu a chybového výstupu [5]

Jedním z aspektů, které byly kladeny při návrhu démona bylo jeho automatické spouštění. Většina démonů je automaticky spouštěna při startu operačního systému, popřípadě v návaznosti na jiný program, který je spouštěn během startu systému. V zadání je jasně uvedeno, že program musí být spouštěn po startu *display manageru*.

5.2 Komunikace s démonem

Abychom mohli démonovi předávat určité příkazy je nutné určit jak. Jako nejvhodnější způsob mi přišlo využít *POSIX message queues*[1]. Práce je obdobná práci se soubory, ale narozdíl od nich jsou *message queues* přizpůsobeny na výměnu dat mezi procesy. Démon vytvoří frontu zpráv[6] a čeká, jestli do ní někdo zapíše. Klient, který chce komunikovat s démonem, otevře frontu se stejným názvem a může takto posílat zprávy[2]. Velkou výhodou oproti souborům je priorita zpráv. Můžeme tak definovat protokol pomocí čísel priority. Zpráva s prioritou 1 vyjadřuje něco jiného než zpráva s prioritou 2.

Každý klient má vlastní frontu, na které přijímá zprávy od démona. S démonem se pomocí zaslené zprávy dohodnou na jménu fronty a démon je pak informuje o změně stavu monitorů. Pro název takovéto fronty mi přišlo vhodné použít číslo procesu klienta, které je pro každý proces unikátní.

5.3 Uložení konfigurací

Program musí mít přehled o tom, jaká konfigurace je aktuálně používaná, kolik monitorů je připojených a jak nastavených. Zároveň musí být schopen porovnávat tuto konfiguraci

[1] Proces číslo 1, pomocí něho se vytváří další procesy.
[2] Vytváření a otevírání souborů musí být nezavislé na rodiči procesu.
[3] Při vytváření dalších podprocesů musí být démon rodičem.
[4] Aby bylo zajištěna nezávislost na připojených discích.
[5] Následně je možné přesměrovat chybový výstup na logovací soubor.
[6] Message queues bychom mohli volně přeložit jako fronty zpráv.

s konfiguracemi uloženými uživatelem. Jako nejlepší varianta mi připadlo využít formátu XML[4]. Výhodou je, že existují knihovny snadno dostupné v linuxových distribucích, které umožňují snadno s XML pracovat. Zvolil jsem proto knihovnu *libxml2*, která je zaštítěna *The GNOME Project*, a která umožnuje parsovat, číst a vytvářet XML soubory. Jediným nedostatkem je, že dovoluje čtení pouze v jednom směru. Pro využívání knihovny programem je nutné nainstalovat balíčky `libxml2` a `libxml2-dev`. Při překladu, pak překládat podle následujícího vzoru:

```
gcc -o example ‘xml2-config --cflags‘ example.c ‘xml2-config --libs‘
```
[10]

5.4 Klient

Klient byl při návrhu definován jako grafický klient napsaný v Qt a C++. Při implementaci jsem z toho také vycházel. Při testování bylo vhodné používat i klienta z příkazové řádky, proto vznikl i tento klient. Důležitým faktorem při vytváření konceptu grafického klienta bylo, aby umožňoval nastavení, které *normální* programy nenabízí, ale tyto funkce jsou dostupné v knihovně XRandR – to znamená například zmiňovaný scaling a panning. Pojmem *normální* je myšleno zabudované v nejrozšířenějších grafických nástavbách linuxu, tj. GNOME a KDE, popř. Unity.

Kapitola 6

Konfigurace v XML

Navržená konfigurace se skládá ze dvou částí, první část je označeno tagem *session* a nacházejí se zde informace o sestavě *screen*, *output* a monitorů. V druhé části je uvedena konfigurace a nastavení připojených monitorů a obrazovek.

6.1 Část session

V této části je uložena informace o počtu *screen* a počtu monitorů. Tyto informace jsou převážně pro urychlení a kontrolu. Další informací jsou jednotlivé *screen*, jako atribut jsou uváděny počty *outputů*, kontrolerů a hlavně číselný identifikátor, pod kterým je *screen* v X serveru uložena. Ten je důležitý pro jednoznačnou identifikaci.

V sekci *screen* je uveden výstup s atributem, že je připojený nebo že je odpojený a také je uvedeno pojmenování výstupu. Pokud je výstup připojen, jsou zde uvedeny informace o monitoru, který je na tento výstup připojen. Důležitým faktorem je EDID, pomocí něhož můžeme říct, jestli je připojen stejný monitor. U výstupu jsou napsané módy, které monitor hlásí, že je může používat. Pro snažší počítání je zde obnovovací frekvence uvedena nejen číselně, ale také jako jednotlivé její složky – `dotClock`, `vTotal` a `hTotal`. Výslednou obnovovací frekvenci pak spočítame pomocí vzorečku `dotClock / (hTotal * vTotal)`.

6.2 Část config

Pod tagem *config* se nachází konfigurace monitorů uvedených v části *session*. Tento tag má ještě speciální atribut `fit`, který definuje, jak musí aktuální kofigurace s tímto nastavením souhlasit, aby bylo přijato. Pro hladký běh je nejlepší uvádět číslo 31, které znamená binárně 11111. Jedná se o návrh vyhodnocování konfiguračních skriptů. Při porovnávání částí *session* dojde k vyhodnocení následujících aspektů:

1. Jsou uvedeny stejné EDID monitorů
2. Monitory mají stejné módy
3. Je připojen stejný počet monitorů
4. `screen` mají stejné atributy (počet výstupů, počet kontrolerů a unikátní číslo)
5. Je připojen stejný počet `screen`

Démon vyhodnocuje pouze ty, u kterých souhlasí všechny zmiňované pravidla. Bylo by však možné napsat takové nastavení u kterého nezaleží na EDID monitorů, ale zaleží na jejich módech, pak by konfigurace měla číslo 30. Vypsané aspekty odpovídají jednomu binárnímu číslu, EDID je 1, módy jsou 2 atd. Při spouštění démona, lze nastavit jaké nejhůře vyhovující konfigurace má příjímat. Jedná se ale o experimentální část a je doporučeno používat pouze `fit='31'`.

Pod nastavením se skrývá výpis všech *screen*, u kterých je uvedena celková šířka a délka v pixelech, ale i v milimetrech. Také je zde uvedena maximální šířka a výška v pixelech, které dovoluje *screen* nabývat. Každá `screen` má v sobě uvedeny výstupy a jejich konfigurace. Výpisem těchto konfigurací zjistíme, co všechno se dá u monitorů nastavit.

connected Nabývá hodnoty `true` nebo `false`, podle toho, jestli je k výstupu připojený monitor.

crtc Popisuje, jestli je monitor aktivní, což znamená jestli je k němu připojený kontroler.

name Jméno výstupu. Například `VGA1`.

vTotal,hTotal,dotClock Informace k výpočtu obnovovací frekvence.[1]

width,height Šířka a výška *panningu*, pokud jsou obě hodnoty nula je vypnutý.

rotation Rotace a zrcadlení, hodnota je uvedena decimálně, ale pro rozlišení rotace a zrcadlení je nutné je převést do binární soustavy.

scaleX,scaleY Zvětšení zobrazení, tzv. `scaling`. Hodnoty 1.0 jsou výchozí.

mode Rozlišení módu.

crtcX,crtcY Pozice kontroleru oproti `framebufferu`. Viz sekce 4.1.

[1] Obnovovací frekvence je vždy spjata s módem

```
<?xml version='1.0'?>
<xrandr>
  <session>
    <NumberScreens>1</NumberScreens>
    <Screen outputNumber='4' crtcsNumber='2' id='0'>
      <output crtc='yes' connected='true' name='LVDS1'>
        <EDID>
          00ffffffffffff0006af2c3200000000
          00130103801d10780a15859758538a26
          2550540000000101010101010101010101
          010101010101261b5664500016303020
          360025a41000001800000000f00000000
          000000000000000000020000000fe0041
          554f0a2020202020202020202000000fe
          00423133335857303320563220 0a0049
        </EDID>
        <mode name='1366x768' prefered='yes' vTotal='790' hTotal='1466'
          dotClock='69500000' refreshRate='60.0100'/>
        <mode name='1360x768' vTotal='798' hTotal='1776'
          dotClock='84750000' refreshRate='59.7990'/>
      </output>
      <output connected='false' name='VGA1'> </output>
    </Screen>
    <NumberMonitors>1</NumberMonitors>
  </session>
  <config fit='31'>
    <Screen id='0' heightMM='361' widthMM='203' height='768' width='1366'
     heightMax='8192' widthMax='8192'>
      <output crtc='yes' connected='true' name='LVDS1' vTotal='790' hTotal='1466'
         dotClock='69500000' height='0' width='0' rotation='1' top='0' left='0'
         scaleY='1.0' scaleX='1.0' mode='1366x768' crtcY='0' crtcX='0'/>
      <output connected='false' name='VGA1'/>
    </Screen>
  </config>
</xrandr>
```

Příklad 6.1: Konfigurace uložená v XML

Kapitola 7

Definovaný protokol

Komunikaci mezi démonem a klientem by se dalo přirovnat k práci s iterativním serverem a vztahu server a klient. Na základě zpráv, které *server* dostává, vykonává práci, která byla na tyto zprávy definována. Této komunikaci a definici bychom mohli říkat protokol. Jako prostředek přenosu je použit již zmiňovaný *POSIX message queues*[1]. Jako výchozí komunikační klient je fronta zpráv s označením `/xxrr`. Jakákoliv schránka musí začínat symbolem `/`, dále následuje její název. Tyto zprávy jsou používány jako soubory, ale nejsou běžně nikde připojeny. Lze však soubory s těmito zprávami připojit do složky. Pro kontrolu je tedy možné použít následující příkazy a prozkoumat, jestli soubory se zprávami existují.

```
xklang00@merlin: ~$ mkdir /dev/mqueue

xklang00@merlin: ~$ mount -t mqueue none /dev/mqueue[1]
```

Také je důležité brát ohled, jestli ostatní programy mají právo zapisovat do těchto souborů, aby se informace dostaly k serveru. Ale jelikož se jedná de facto o soubor, můžeme změnit oprávnění souboru pomocí funkce `fchmod`.

Zprávy uložené do `/xxrr` jsou příchozí informace pro server. Server posílá i zprávy, jestli se daný úkon povedl a zda nenastala nějaká chyba. Server odesílá tyto informace na frontu, kterou si klient zaregistruje. Jméno této fronty je tvořeno pomocí PID procesu a vypadá například takto `/xxrrclient1234`.

Server přijímá a rozpoznává 7 rozdílných zpráv. Priorita je u těchto zpráv velice důležitá, server takto rozpozná o jaký pokyn se jedná. Priority s vyšším číslem jsou vyhodnocovány dříve. Uvedené zprávy jsou ohodnocené prioritou 7 až 1.

Zrušení registrace klienta Poslaná zpráva ukončuje posílání zpráv od serveru klientovi.

Registrace klienta Poslaná zpráva zajišťuje registraci klienta pro příjem zpráv. V textu zprávu musí být obsažen název fronty zpráv pro komunikaci s klientem.

Vložení konfigurace 1. část Po zprávě s touto prioritou musí následovat zpráva s prioritou **1**. Tato zpráva nese informaci o tom, jaký soubor se má překopírovat do adresáře konfigurací. V této zprávě musí být uvedena absolutní cesta k souboru.

Znovunačtení Zpráva s touto prioritou zapříčíní, že démon provede přepsání aktuální konfigurace do XML souboru, ze kterého vychází při porovnání.

[1] Viz sekce 5.2

Rekonfigurace Zpráva způsobí, že server projde všechny uložené konfigurace a vybere nejvhodnější. Toto je vhodné použít při uložení nové konfigurace, abychom tak docílili načtení konfigurace.

Vymazání konfigurace Podle názvu souboru vymažeme konfiguraci z adresáře.

Uložení aktuální konfigurace Monitory můžeme nastavit v jakémkoliv programu a pak tuto sestavu pouze uložit. Ve zprávě musí být uvedeno jméno, jak bude konfigurace uložena.

Vložení konfigurace 2. část Před touto zprávou musí být serveru zaslána zpráva **Vložení konfigurace 1. část**. Zpráva obsahuje jméno, jak bude konfigurace uvedena pod cestou z první části zprávy.

Klient rozpoznává 3 informace. Jedná se o informace, které posílá server na zpátek klientům. Klient může fungovat i bez těchto zpráv, ale nebude informovaný, jestli nenastala nějaká chyba nebo vše dopadlo správně. Klient, který je napsán pro příkazovou řádku tyto zprávy nepříjímá. Server čeká a přijímá zprávy a zároveň sleduje, jestli se nezměnila konfigurace a není tak nutné klienta o této události uvědomit. Proto démon vytváří nové vlákno, ve kterém sleduje změnu výstupů, to znamená, jesli nenastala změna nastavení a zároveň se nezměnil počet monitorů. Uvedené zprávy jsou ohodnocené prioritou 3 až 1.

Chybové hlášení Zpráva znamená, že serveru se nepovedlo provést příkaz, který mu byl poslán. Ve zprávě je uveden soubor, který se nepovedlo aplikovat a který byl pro hladký běh programu vymazán.

Vše proběhlo v pořádku Pokud se serveru povedlo provést příkaz, pošle tento signál.

Změna konfigurace Signál poslaný s tímto číslem znamená, že nastala změna sestavy monitorů. Některý monitor se připojil a je nutné, aby klient provedl znovunačtení.

Kapitola 8

Automatické spouštění

V zadání mé bakalářské práce stojí, že „*Řešení musí umět při startu manažeru obrazovky detekovat připojené monitory*“. Tím, že se jedná o démona, již máme určeno, že bude samostatný a že bude běžet tzv. na pozadí. Musíme také zajistit, že bude automaticky spouštěn. Základní vlastností většiny linuxových distribucí je, že se po načtení jádra spustí *init* démon. Tento démon spouští ostatní démony klasicky v závislosti na *runlevel*. Těchto levelů je více a na základě jeho čísla se spustí příslušné skripty na spouštění dalších programů. Jeden bývá vyhrazen na vypnutí, kdy jsou programy korektně ukončeny pomocí skriptů a další level je na restart počítače.

Některé moderní linuxové distribuce tento způsob považují za zastaralý a vytvořily tak náhrady a vylepšení. Mezi dva nejrozšířenější patří *upstart* a *systemd*. Jedná se o náhradu klasického *init* a použití modernějšího způsobu, jak spouštět další služby. Největší výhodou těchto způsobů je návaznost. Každý skript při spuštění může vysílat signál, kterým dává vědět, že je již spuštěn. Díky čemuž můžou ostatní inicializační skripty navázat svojí činností. Velkou nevýhodu spatřuji v tříštění již tak dost roztříštěného pole linuxově založených operačních systémů.

Rozhodl jsem se využít, co nabízí distribuce Ubuntu, protože ji považuji za poměrně rozšířenou a známou mezi uživateli počítačů. Je to pouze můj subjektivní pohled. Zvolil jsem podle toho preference programu a zrealizoval program tak, aby byl spustitelný především na aktuální distribuci Ubuntu. Z toho vyplývá, že jsem se rozhodnul využít možností co nabízí *upstart*. Není problém doinstalovat *upstart* i do jiných distribucí, otázkou zůstává, jestli je to nutné, když jiné distribuce využívají odlišných komponent.

8.1 Upstart

Upstart[7] je událostně založená náhrada pro `/sbin/init` démona, která se stará o služby a úlohy během bootování, zastavování během vypínání a spravování za běhu systému. Upstart umožňuje spouštět a zastavovat služby podle událostí, signálů. Tyto signály dokáže generovat a odesílat. Tyto služby mohou být také znovu obnoveny při náhlém pádu. Komunikace mezi démony probíhá pomocí D-Bus. Všechny konfigurační skripty démonů jsou uloženy v adresáři `/etc/init/`, můžeme tak snadno dohledat čím a jak se spouští jednotlivé programy. Například pro nás je důležitý *display manager*. Výchozí správce je v Ubuntu program *lightdm*, nebo-li odlehčený *display manager*. Jeho konfigurace pro *upstart* se nachází v souboru `/etc/init/lightdm.conf`. Při startu tohoto skriptu se pošle signál `login-session-start`, na který můžeme navázat náš vlastní skript.

Skript psaný pro *upstart* má několik sekcí. Na začátku je důležité definovat, kdy se skript spustí. Můžeme vycházet z klasických *runlevelů* nebo použít některý ze signálů, které skripty tvoří. Náš program se má spouštět při startu manažera obrazovky, který zasílá zmíněný signál. Náš skript bude ukončen při restartování a standardním vypnutím, začátek našeho konfiguračního skriptu bude vypadat takto:

```
start on login-session-start

stop on runlevel [016]
```

Dále může následovat samotný skript, který je ohraničen klíčovými slovy `script` a `end script`. V této části voláme potřebného démona. Je zde také možnost definovat, co se vykoná před startem `pre-start script` a před zastavením `pre-stop script`. Programy takto spuštěné mají práva root, ale do adresáře `/etc/init/` může také zapisovat pouze uživatel s právy uživatele root.

Při spouštění našeho programu ale nastává mírný problém. Program je grafický a závisí tedy na X serveru. Ten po naběhnutí *manageru* je již spuštěn, ale neznamená to, že k němu máme přístup. Přístup k X serveru je řešen pomocí *Xauthority*, které manažer vytváří a umožňuje tak přístup ke grafickým prvkům. Pro *lightdm* se tento soubor nachází `/var/lib/lightdm/.Xauthority`, ale po přihlášení uživatele je smazán a soubor se nachází pro jednotlivého uživatele v jeho složce, tedy `/home/`*uživatel*`/.Xauthority`. Museli bychom proto po přihlášení zjišťovat, kdo je přihlášený. Další možností je, že tyto autority jsou uloženy i v „pamětí“ systému, v dočasných souborech a zůstavají na stejném místě i po přihlášení uživatele. Tento soubor se pro *lightdm* a *gdm* nachází `/run/lightdm/root/`, ale pro *kdm* ho nalezneme zde `//var/run/xauth/`. Pokud program nemá přístup k *display*, tak se pokusí najít jeden z těchto souborů a dostat tak autorizovaný přístup k X serveru. Abych byl přesnější, je nutné v běhovém prostředí definovat proměnnou `XAUTHORITY`. Také je potřeba definovat `DISPLAY`, ten je nastaven na výchozí hodnotu `:0`, což znamená první *display*. V případě nutnosti, je potřeba tuto hodnotu změnit v `/etc/init/xxrr.conf`, kde se nachází po instalaci spouštěcí skript pro *upstart*.

```
start on login-session-start
stop on runlevel [016]

script
  export DISPLAY=:0
  export XAUTHORITY=/var/lib/lightdm/.Xauthority
  exec su -c /usr/bin/xxrr
end script
```

Příklad 8.1: Upstart skript

Kapitola 9

Klientská část

Uživatel nepotřebuje vědět, jak funguje démon, ani jaké zprávy jsou nebo nejsou povoleny. Nahlíží na program z klientské strany. Kvůli zřejmé roztříštěnosti je programování grafického programu v GNU obtížné. Jediná věc, která spojuje všechny grafické nástavby je příkazový řádek. Možnost spustit si terminál a obsluhovat jej nabízí snad každá linuxová distribuce. Klient pro příkazovou řádku jsem původně vůbec neměl v úmyslu, ale při porovnání, co dokáže program *xrandr* a jeho grafické nástavby, jsem volil podobnou strukturu.

Důležitou vlastností klienta určeného pro terminál je uložení aktuální konfigurace. Můžeme si tak libovolně nastavit monitory, například pomocí *xrandr*, a následně přes `xxrr -s example.xml` uložení této konfigurace k démonovi. Není potřeba, aby klient běžel v grafickém režimu. Je tedy možné ovládat démona i vzdáleně, například přes *ssh* bez použití X serveru. Při vytváření vlastní konfigurace je nutné dbát na validní hodnoty v xml souboru. V případě, že nastane problém s vyhodnocováním konfigurace, bude taková konfigurace smazána, aby se předešlo problémům.

V přímém srovnání s grafickým klientem nabízí *CLI*[1] mnohem více možností. Podporuje téměř všechny zprávy, které server umožňuje přijímat. Nenabízí přehled nad výsledkem operace. Je nutné sledovat logovací soubory ručně. Je nutné zobrazit si konfiguraci a vědět, co se v ní nachází a z jakého důvodu. Jedná se tedy o možnosti pro zkušenější uživatele, kteří ví, jak můj program funguje a potřebují pouze dát jasný pokyn. Pro větší jazykovou dostupnost jsem zvolil anglický jazyk pro nápovědy a texty.

```
usage: xxrrcli [-h] [-d file] [-a file] [-s] [-r]
  -d [file]
delete [file] with configuration from /etc/xxrr/xmls
  -a [file] [name]
add [file] to directory /etc/xxrr/xmls and save as [name]
  -s [name]
add actual configuration as [name] to /etc/xxrr/xmls
```

Příklad 9.1: Část nápovědy klienta pro příkazovou řádku

[1] Command Line Interface, rozhraní pro příkazovou řádku

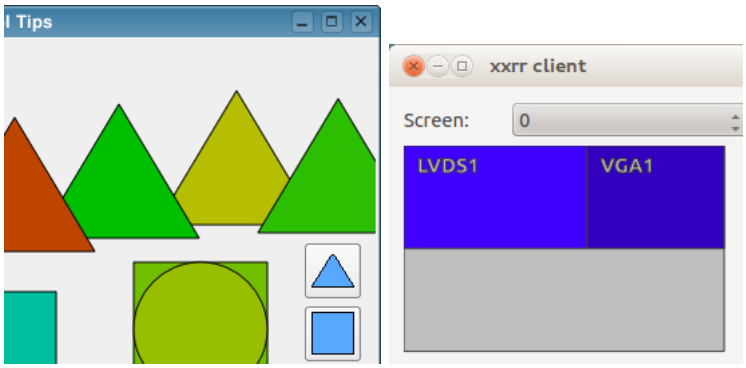
9.1 Grafický klient a Qt

Při tvoření grafického klienta jsem použil framework *Qt* a jazyk *C++*. Pro programování jsem pak využil nástroje *Qt Creator*, který je volně dostupný. *Qt* jsem zvolil z důvodu, že je dle mého lépe zdokumentované, než například konkurenční *GTK+*. *Qt* také staví od základu na *C++* narozdíl od *GTK+*, které je psané v jazyce *C*. Další zajímavou věcí je IDE[2], které nese název *Qt Creator*. Toto vývojové prostředí není ideální, ale nabízí ucelený nástroj pro vývoj projektů, kompilaci, design i debug. Je taktéž multiplatformní, tedy se stejným prostředím se můžeme setkat také ve Windows.

Použití *Qt* také usnadnil fakt, že prostředí *Unity 2D*, které je jedním z výchozích prostředí použitých v Ubuntu staví na *Qt* a není tak třeba do Ubuntu doinstalovávat potřebné knihovny. Dalším příjemným bonusem je množství příkladů, které jsou dostupné. Tyto příklady jsou vhodné pro začátek a pro inspiraci. Jsou šířené pod licencí BSD, která je dostatečně svobodná a dají se díky ní tvořit dále libovolně licencované programy.

Nejdůležitější bylo načtení konfigurace, to znamená přečtení XML a načtení informací do objektů. Ke správnému uložení této sestavy jsem využil třídy `QXmlStreamReader`, která funguje velice obdobně jako zmiňovaná `libxml2`. Sekvečně prochází soubor a prohledává, jestli jsou obsažené důležité tagy. Jednotlivé *output* a informace o nich jsou naskládány do objektu `QList`, který slouží jako obousměrně zřetězený seznam. Tento seznam je přiřazen jednotlivým *screen*. Tyto obrazovky jsou opět svázány, vznikne tak struktura objektů odpovídající struktuře XML a její části *session*. Poté je prohlédávána sekce *config*, kde jsou uchovány aktuální nastavení. Získáme tak všechny informace z aktuální konfigurace.

Jedním z příkladů, které jsem využil, je demo Tooltips. Tooltips umožňuje po kliknutí tlačítka vykreslit kruh, čtverec nebo trojúhelník. Po kliknutí na některý z objektů je možné jej přesouvat. Toho jsem využil pro vykreslování monitorů, které jsou v dané konfiguraci. Bylo nutné definovat, kam jde posunout čtverec znázorňující monitor a na jakém monitoru je aktuálně kliknuto. To jsem odlišil rozdílným odstínem modré. Do vykreslovaných monitorů jsem přidal názvy, které je jednoznačně odlišují i v rámci X serveru. Šedé pole pak označuje plochu, do které mohou být monitory poskládány. Rozdíl příkladu a mého použití můžete vidět na následujícím obrázku.



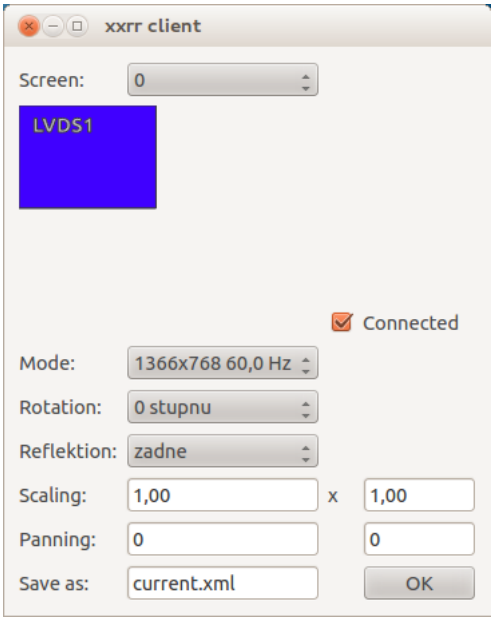
Obrázek 9.1: Využití dema Tooltips k zobrazení monitorů

Dále se program skládá z několika `QComboBox`, neboli výběrových polí a také `QlineEdit` políček, do kterých se může psát. Hodnoty jsou do nich vkládány podle aktuálního monitoru a podle dat uvedených v *config* načtené konfigurace. První možností je volba plochy. Tím se

[2]Integrated Development Environment, vývojové prostředí

program odlišuje od většiny grafických programů na nastavení monitorů, protože v závislosti na jaké *screen* je program spuštěn, je možné kofigurovat monitory přiřazené této *screen*. Ale tím, že komunikace probíhá přes démona, je možné nastavovat libovolnou plochu.

Podle zvolené plochy se vykreslí monitory, které jsou pak možné nastavovat. U těchto výstupů můžeme nastavovat pozici monitorů pohybováním, posouváním ve stylu `drag&drop`, dalé pak nastavovat jejich mód, rotaci, zrcadlení, scaling a panning. Také samozřejmě můžeme nastavit, jestli je monitor aktivní nebo zůstane zapojený, ale nepřiřadí se mu kontroler.



Obrázek 9.2: Grafický klient s nastavením jednoho monitoru

Posledním políčkem je název souboru, které určuje, kde bude konfigurace uložena. Je to důležité, protože démon vyhodnocuje soubory na základě názvu a zároveň jak vyhovuje. Znamená to, že konfigurační soubor s názvem abecedně výš bude upřednostněn před tím, který je abecedně níž. Dále je dobré pojmenovávat tato nastavení logicky, podle připojených monitorů, abychom mohli konfigurace spravovat a vědět, jaké máme uložené.

Zároveň s grafickou stránkou je v programu vytvořeno další vlákno, které na začátku zaregistruje u démona frontu zpráv a čeká, jestli od něj nepříjde zpráva k rekonfiguraci. Také po odeslání konfigurace je pomocí tohoto vlákna řešeno potvrzení o přijetí, případné vypsání chybové hlášky.

V přímém porovnání s druhým klientem nenabízí grafický klient tolik možností práce se serverem, ale nabízí rozšířené spektrum nastavení oproti výchozím nastavením v běžných grafických prostředích.

9.2 Tvorba deb balíku

Balíkovací systém je rozšířen v celém spektru linuxu a jeho distribucí. Jedná se o snadný a automatický způsob, jak instalovat, aktualizovat, konfigurovat a odstraňovat software.[15] Balíček obsahuje program, který si můžeme nainstalovat. Také je zde uchovávaná verze programu, takže se dá rozhodnout, jestli je balík novější či nikoliv. Zároveň jsou v informacích v balíčku uvedené i závislosti, pokud tedy program využívá nějakou knihovnu, zaznamená

programátor jakou a balíček nejde nainstalovat, dokud daná knihovna není nainstalována. Je tak postaráno o to, aby program mohl správně fungovat.

Bohužel i zde se potýkáme s roztříštěností linuxových distribucí. Distribuce založené na Debianu, tzn. Ubuntu aj., používají balíkovací systém *deb* oproti distribucím založených na Red Hatu, tzn. Fedora, CentOS aj., které používají *rpm*. Tyto balíkovací systémy jsou nejrozšířenější, ale existují i jiné, které používají například Gentoo Linux nebo Arch Linux.

Protože jsem optimalizoval program na distribuci Ubuntu, využil jsem balíkovacího systému *deb*. Postupoval jsem podle návodu na wiki stránkách Ubuntu Česko[18]. Je potřeba vytvořit adresářovou strukturu, která odpovídá výsledku instalace, tzn. kam se jednotlivé soubory programu zapíší. V našem případě bude nutné nainstalovat skript `xxrr.conf` pro spouštění démona přes `upstart` do složky `/etc/init/`. Dále pak nainstalovat spustitelné soubory programu do složky `/usr/bin` a to sice klienty i démona, tzn. `xxrrd`, `xxrrcli` a `xxrrqt`. Také vytváříme adresář `/etc/xxrr` a `/etc/xxrr/xmls` pro konfigurační skripty démona.

```
DEBIAN/control
       md5sums
etc/init/xxrr.conf
etc/xxrr/xmls/
usr/bin/xxrrd
        xxrrcli
        xxrrqt
```

Příklad 9.2: Adresářová struktura pro tvorbu balíčku

Po vytvoření struktury je zapotřebí vyplnit soubor o metadatech. Pro tento soubor je nutné vytvořit adresář `DEBIAN`. Soubor nese název `control` a je zde třeba uvést název balíčku, verzi, závislosti, architekturu[3], instalovanou velikost, kontakt a jméno na správce balíčku a popis.

Druhým souborem ve složce `DEBIAN` je md5sums, kde se nachází kontrolní součty jednotlivých souborů v balíčku. Součty můžeme získat příkazem:

```
find * -type f ! -regex '^DEBIAN/.*' -exec md5sum {} \; > DEBIAN/md5sums
```

Na závěr musíme všem souborům v balíčku změnit přístupová práva na uživatele root:

```
sudo chown -hR root:root *
```

Následně můžeme vytvořit balíček za pomoci příkazu

```
sudo dpkg-deb -b tmp xxrr_0.1_amd64.deb
```

Balíček je hotový a dá se na cílovém počítači nainstalovat příkazem

```
sudo dpkg -i xxrr_0.1_amd64.deb
```

[3]Zda se jedná o aplikaci pro `amd64`, `i386` nebo na tom nezáleží.

Kapitola 10

Testování

Aplikace byla zevrubně testována na dvou strojích. Aby byla zajištěna variabilita grafických karet, byl jeden z počítačů osazen grafickou kartou ATI X1300 a zároveň procesorem Sandy Bridge s integrovanou grafickou kartou od firmy Intel. K tomuto stolnímu počítači bylo možné připojit až čtyři monitory tak, že na každé grafice byly připojené dva monitory. Jako operační systém byl nainstalován Ubuntu 11.10. Problém u tohoto počítače se nacházel v nastavení dedikované grafické karty, protože s ovladačemi přímo od společnosti ATI karta vůbec nefungovala. Bylo nutné přes `Xorg.conf` nastavit svobodné komunitní ovladače *radeon*. Díky dvojici grafických karet vznikla také dvojice *screen*.

Druhý počítač na testování byl notebook, který taktéž obsahoval integrovanou grafiku v procesoru od Intelu, ale také grafickou kartu od firmy Nvidia s technologií Nvidia Optimus. Tato technologie zajišťuje přepínání těchto grafických karet za běhu operačního systému. Problém je opět v ovladačích, protože zatímco ve Windows je dodaný software funkční, v linuxu nic od výrobce nenalezneme. Musíme buďto používat po dobu běhu X serveru pouze jednu z grafických karet a nebo využít experimentálních ovladačů vytvořených komunitou. Pro snadnější chod jsem proto zvolil pouze grafickou kartu od Intelu. Čas od času ale došlo k přejmenování výstupů[1], které grafické karty mají společné a tak může dojít k nevyužití konfiguračních profilů. Ještě jednou nevýhodou takto spojených grafických karet je, že mají i společné *crtc*, takže i když má notebook dva výstupy na monitory, není možné využít třech monitorů. Na tomto počítači byl využíváno Ubuntu 12.04 od alfa verze až po stabilní verzi.

Byly testovány různé konfigurace, zrcadlení podle osy x i y, rotace, panning, scaling. U posledního zmiňovaného se stalo, že by měla být použitelná jakákoliv transformační matice. Povedlo se ale zprovoznit pouze zvětšení a zmenšení. To znamená matice 3×3, která mimo diagonálu má samé nuly a na pozici $[1, 1]$ má hodnotu zvětšení v ose x a na pozici $[2, 2]$ má hodnotu zvětšení v ose y. Na pozici $[3, 3]$ má číslo 1. Abych zajistil větší kompatibilitu, byla omezená funkčnost. Rotace po 90 stupních se dá navíc zajistit za pomocí hodnot rotace.

Postupně byly také upravovány atributy v XML formátu konfigurací. Hodnota obnovovací frekvence byla použita v atomické podobě s `dotClock`, `vTotal` a `hTotal`. Podobně u tagu *screen* přibylo uchovávání maximální možné šířky a výšky v pixelech.

Během testování bylo zapracováno hlášení chyb a logů. Logovací soubor nalezneme v `/etc/xxrr/` jakožto `xxrrd.log`. Při nalezení chyby je také podána zpráva grafickým klientům. V případě, že některý z konfiguračních protokolů nemůže být zpracován, je smazán. Tak je zajištěno, že jsou uchovávány pouze funkční sestavy nastavení. I v tomto případě je

[1] Z `VGA1` se výstup přejmenoval na `VGA2` a zpět.

informace o vymazání napsaná do logovacího souboru.

U příjímání a posílání zpráv bylo chybou, že grafická aplikace soubor na přijímání obsahu vytvořila a při ukončení uzavřela, avšak soubor je potřeba při ukončení vymazat, protože počet otevřených front zpráv je omezený. Jde toto omezení zvětšit, ale výchozí hodnota je nastavena na 256 otevřených front, což by při běžném používání mělo postačovat.

Testování automatického přihlašování proběhlo na *lightdm*, *kdm*, *gdm* a *xdm*. Testováno bylo také na některých počítačích mých spolužáků. Především však testy probíhaly v laboratoři biometrických systému v místnosti S215.

Kapitola 11

Závěr

Ze zadání plyne, že mým úkolem bylo nastudovat možnosti detekce a správy monitorů v linuxu, poté navrhnout a implementovat takové řešení, které bude monitory detekovat, spravovat a konfigurovat. Prostudoval jsem možnosti, které se mi nabízely a zvolil jsem jako nejvhodnější prostředí pro správu X server a jeho rozšíření XRandR. S těmito prostředky byl vytvořen démon, který se o konfigurace stará a umožňuje jejich spuštění a uplatnění. K démonovi jsem vytvořil grafického klienta, který umožňuje upravit aktuální konfiguraci a předložit ji démonovi k načtení. Zároveň byl vytvořen i odlehčený klient pro příkazovou řádku. Zdrojové kódy i sestavené balíčky se dají stáhnout na následující adrese:

`http://eva.fit.vutbr.cz/~xklang00/IBP/`

Také jsou tyto soubory obsaženy na DVD přibalené k této práci. Všechny zdrojové kódy jsou psané pod licencí BSD.

Díky této práce jsem rozšířil znalosti v oblasti grafických nástaveb linuxu, tj. například X server. Zajímavé bylo sledovat vývoj tohoto programu v čase, přestože jeho návrh a začátek byl již před 30 let, nadále dominuje v linuxovém světě jako nejznámější grafická nástavba. Zajímavé je také kolik toho umí XRandR. Vlastnosti typu scaling a panning nejsou moc často vidět a ještě méně, pokud člověk chce nastavit tyto věci graficky. Xrandr v příkazové řádce má mnoho funkcí a uživatelé o tom ani nemusí vědět.

Ve své bakalářské práci jsem se zaměřil převážně na praktičnost mého úsilí, aby uživatelé, kteří by si můj program nainstalovali, jej považovali za využitelný v praxi. Jde také o to, aby jej chtěli používat a ukládali své konfigurace. Při testech jsem proto také pozoroval, jestli je program použitelný například i na notebooku. Otestoval jsem také rozdílné hardwarové kombinace, abych zajistil variabilitu. Tato variabilita nebyla bohužel zajištěna v rámci linuxové distribuce, protože jsem se rozhodnul program optimalizovat pro Ubuntu. Překvapila mě roztříštěnost operačních systémů založených na linuxu. V rámci možností jsem volil vhodné programové nástroje. Pěkným příkladem je automatické spouštění pomocí *Upstart*. Ubuntu používá jiného správce démonů než například Fedora nebo openSUSE. Stejný problém nastává i u balíkovacího systému nebo manažera obrazovky.

V programu by bylo dobré pokračovat i po ukončení bakalářské práce. Program považuji za smysluplný a praktický a rád ho budu do budoucna používat, udržovat ho a vyvíjet, aby vyhovoval přáním uživatelů. Například grafický program sice je napsán, aby byl použitelný, ale pohled informatika a uživatele bývá často odlišný. Také by bylo vhodné napsat podobného klienta i v jiném grafickém prostředí, například v konkurenčním *GTK+*.

Jednotlivé možnosti jsem shrnul do několika bodů.

- Přívětivější uživatelské rozhraní
- Rozšířit aplikaci mezi uživatele
- Upravit program, aby byl spustitelný i v jiných distribucích, než Ubuntu
  - Místo *upstart* využít i *systemd*
  - Jiný balíkovací system
  - ...
- Napsat grafického klienta vhodného pro prostředí GNOME, případně jiné grafické nástavby X serveru[1]

[1]Grafický klient je psaný v *Qt* použitelný především v KDE

Literatura

[1] mq_overview(7) - Linux manual page. [online], [cit. 2012-01-31].
URL <http://www.kernel.org/doc/man-pages/online/pages/man7/mq_overview.7.html>

[2] POSIX.4 Message Queues. [online], [cit. 2012-01-31].
URL <http://www.users.pjwstk.edu.pl/~jms/qnx/help/watcom/clibref/mq_overview.html>

[3] xorg.conf. [online], [cit. 2012-03-16].
URL <http://www.x.org/releases/current/doc/man/man5/xorg.conf.5.xhtml>

[4] Extensible Markup Language (XML) 1.0 (Fifth Edition). [online], [cit. 2012-04-30].
URL <http://www.w3.org/TR/REC-xml/>

[5] ArchWiki: Xorg (Česky) – ArchWiki. [online], [cit. 2012-05-09].
URL <{https://wiki.archlinux.org/index.php/Xorg_(%C4%8Cesky)}>

[6] Gettys, J.; Packard, K.: *The X Resize, Rotate and Reflect Extension.* [cit. 2012-01-28].
URL <http://www.x.org/releases/X11R7.6/doc/randrproto/randrproto.txt>

[7] Ltd., C.: upstart - event-based init deamon. [online], [cit. 2012-05-05].
URL <http://upstart.ubuntu.com/>

[8] Scheifler, R. W.; Gettys, J.: *X Window System: Core and extension protocols, X version 11, releases 6 and 6.1.* Digital Press, 1996, ISBN 1-55558-148-X.

[9] Tyler, C.: *X Power Tools.* O'Reilly Series, Oreilly, 2008, ISBN 9780596101954.
URL <http://books.google.cz/books?id=V1ZBeNJIx7UC>

[10] Veillard, D.: Libxml2 set of examples. [online], [cit. 2012-04-30].
URL <http://xmlsoft.org/examples/index.html>

[11] Video Electronics Standards Association: *VESA BIOS Extension 3.0.* [cit. 2012-01-25].
URL <http://www.petesqbsite.com/sections/tutorials/tuts/vbe3.pdf>

[12] Watson, D.: Linux Deamon Writing HOWTO. [online], [cit. 2012-01-31].
URL <http://www.netzmafia.de/skripten/unix/linux-daemon-howto.html>

[13] Wikipedia: Extended display identification data — Wikipedia, The Free Encyclopedian. [online], [cit. 2012-01-15].
URL <http://en.wikipedia.org/wiki/Extended_display_identification_data>

[14] Wikipedia: Framebuffer — Wikipedia, The Free Encyclopedian. [online], [cit. 2012-04-30].
URL <http://en.wikipedia.org/wiki/Framebuffer?oldid=202487315>

[15] Wikipedia: Package management system — Wikipedia, The Free Encyclopedian. [online], [cit. 2012-05-09].
URL <http://en.wikipedia.org/wiki/Package_management_system>

[16] Wikipedia: XFree86 — Wikipedia, The Free Encyclopedian. [online], [cit. 2012-05-09].
URL <http://en.wikipedia.org/wiki/XFree86>

[17] Wikipedia: X.Org Server — Wikipedia, The Free Encyclopedian. [online], [cit. 2012-05-09].
URL <http://en.wikipedia.org/wiki/X.Org_Server>

[18] Česko, U.: Vytvoření .deb balíku - Ubuntu Česko. [online], [cit. 2012-05-05].
URL <http://wiki.ubuntu.cz/Vytvo%C5%99en%C3%AD%20.deb%20bal%C3%ADku>

Dodatek A

Obsah DVD

Na přiloženém DVD nalezneme soubor `readme.txt`, který poskytuje nápovědu a manuál, jak program nainstalovat a spustit. Dále zde nalezneme adresář `deb`, ve kterém se nacházají debianové balíčky, binární soubory a složky připravené na vytváření debianového balíčku. V adresáři `sources` se nacházejí zdrojové kódy programu. Poslední adresář je `texsources`, kde jsou zdrojové kódy k LaTeXu a také tato bakalářská práce ve formátu PDF.